下载APP

04 | 容量治理的三板斧：扩容、限流与降级

2021-05-19 吴骏龙

容量保障核心技术与实战 进入课程 >

讲述：吴骏龙
时长 17:17 大小 15.84M

你好，我是吴骏龙。

在前面几讲中，我谈到了容量测试和容量指标分析等工作，这些工作能够帮助你验证服务是否存在容量问题，并定位问题出现的原因。

找到问题以后，我们需要排除这些容量风险，确保服务的稳定性，这就是容量治理需要做的工作了。优秀的容量治理工作不仅仅能**通过一系列手段解决已出现的容量问题，而且还能预防容量隐患，做到防患于未然。**

下面，我们就来认识一下三个常见的容量治理手段：扩容、限流和降级，这三个手段分别基于不同的视角，在容量保障中发挥着各自的价值。在这些内容的讲解中，我会从我经历过的实际案例出发，告诉你一些实践中的难点和坑点，希望能带给你更多的收获。

扩容的实践要点

说到扩容，不得不提到在软件架构设计中经常出现的一个词，叫做“伸缩性”，更高端的说法叫“弹性”。以目前互联网服务的用户体量，已经不太可能通过单台服务器去支撑所有的流量，流行的做法是将多台服务器组成集群统一对外提供服务，伸缩性指的就是能够往这个集群中不断加入新的服务器资源，以应对流量增长的能力，而这个**加入新的服务器的过程，就是扩容**。

如果容量瓶颈发生在服务器资源层面，扩容恐怕是最直接也是最“暴力”的容量提升手段了，尤其是在服务容器化盛行的当下，拉起一个新服务实例，往往只需要秒级时间，扩容的成本得到了有效的降低，能够快速消除容量风险。

我对扩容的态度是：**鼓励将快速扩容作为应急手段，但作为容量治理手段要谨慎，警惕无脑扩容和滥用扩容。**

在实际工作中，很多业务研发一遇到自己的服务容量不足，第一个念头就是扩容，服务本身的优化工作，包括异步处理、读写分离、增加缓存、SQL 调优等等，往往会被忽略，这是有很大风险的。

首先，在开篇词中我就提到过，容量保障是要考虑成本的，如果纯粹依靠扩容去支撑性能低劣的服务，浪费的是大量的资源和金钱。其次，扩容也很容易触碰到边际递减效应，也就是说，服务资源达到一定规模后，再往上扩容的代价会大幅上升。

我在阿里本地生活推动容量风险改进时，说的最多的一句话就是：“不要无脑扩容！”。甚至每到大促活动期间，我都会与全局架构师进行合作，如果有业务方提交的扩容工单中仅仅提到了对扩容资源的需求，而没有考虑任何服务层面的性能优化措施，这样的工单会被直接拒绝。建立谨慎扩容的风气相当重要，如果大量的容量问题都靠“堆机器”去冲摊，服务资源的成本会越来越高。

除了避免无脑扩容，我还要提醒你注意的是，**扩容不能只关注服务本身的资源占用情况，还应同时关注对底层资源的影响**，如：数据库资源、中间件资源等。

举个简单的例子，某服务集群一共有 10 个容器实例，每个实例会建立约 100 个数据库连接，加起来就是约 1000 个连接，假设数据库总共支持的连接数为 1200 个，这是能够支撑现状的。但如果考虑到近期业务增长较快，会导致服务负载较大，需要扩容 5 个实例，

那么总的数据库连接数大约会达到 1500 个，这就肯定支撑不住的，所以对服务进行扩容时，对数据库也需要同步扩容。

再分享一个稍复杂的例子，在微服务体系中有一项服务发现机制，每个服务可以通过该机制获取被调用服务的位置。服务发现机制的一种实现方式是，由一个注册中心建立对每个服务实例的长连接，并维护一个服务状态列表，这样一旦服务状态发生变化，注册中心能够第一时间感知到并对服务状态列表进行更新。

我们很容易想到，这些建立的大量长连接可能会产生瓶颈（主要是消耗内存），当我们对服务进行扩容时，实际上就是增加了服务实例，这会产生更多的长连接。因此在这种服务发现模式下进行扩容时，注册中心的容量也需要同步考虑。

如果你觉得扩容要考虑的点太多，一种通用的做法是，先梳理出被扩容服务的调用链路，看一看流量经过哪些地方，分析一下这些地方会不会受到扩容的影响，再去采取相应措施。

总结一下扩容的实践要点，首先，要建立**服务性能优化胜过简单扩容**的意识，努力通过性能优化腾出容量，避免不经思考直接扩容。其次，扩容要联动系统整体资源共同规划，不能只关注服务资源。

限流的实践要点

扩容的出发点是尽可能提供足够的资源保证服务容量充足，但正如我在第一讲中所提到的，容量保障是需要兼顾成本的，限流就是**在控制成本的情况下对服务容量的一种保护措施**。此外，即便进行了严密的容量规划和系统优化，我们依然无法保证线上流量一定百分百符合既定的预测范围，因为总会有这样那样的突发事件发生，如果流量峰值超过了系统能够承载的极限，相较于紧急扩容，**提前设置合理的限流对系统进行过载保护，是更主动的方式**。

那么，在提前设置限流时，我们应该选择什么样的限流策略，以及在什么位置进行限流呢？下面，我就从限流策略和限流位置两个角度，告诉你在选择策略和位置时应该注意的要点。

首先，从限流策略的角度，可以将常见的策略形象地归纳为“两窗两桶”，分别有：固定窗口、滑动窗口、漏桶算法和令牌桶算法。这些策略本身的实现方式和特点我就不多费笔

墨了，网上已经有很好的学习资料，你可以点击查看。我会将讲解的重点放在如何选择合适的限流策略，以及如何建设有层次的限流体系上。

首先要与你分享的经验是，**限流策略的选择和业务场景应当是高度挂钩的**，不能想当然觉得复杂的策略就一定好。为了让你更深刻的理解这一点，我总结了上述四个限流策略中，与业务场景紧密相关的三大流量控制方式，分别是：流量整形、容忍突发流量和平滑限流，它们的含义如下：

流量整形： 指不管流量到达的速率多么不稳定，在接收流量后，都将其匀速输出的过程，即“乱进齐出”。

容忍突发流量： 指的是限流策略允许流量在短时间内突增，且在突增结束后不会影响后续流量的正常限流。

平滑限流： 指的是在限流周期内流量分布均匀，比如限制 10 秒内请求次数不超过 1000，平滑限流应做到分摊到每秒不超过 100 次请求。反之，不平滑限流有可能在第 1 秒就请求了 1000 次，后面 9 秒无法再发出任何请求。

基于这三个特点，结合不同限流策略对流量控制方式的具体支持情况，我汇总出了下面这张表格。

限流策略	流量整形	容忍突发流量	平滑限流	实现复杂度
固定窗口	不支持	不支持	不支持	低
滑动窗口	不支持	不支持	不支持	中
漏桶算法	支持	不支持	支持	高
令牌桶算法	支持	支持	支持	高

有了这张表格，你就可以根据业务场景去选择合适的限流策略了。比如说，固定窗口无法容忍突发流量，但它实现简单，资源消耗少，如果你的应用服务流量是平缓增长的形态，也没有流量整形的需求，这时采用固定窗口策略进行限流就不失为一种合理又经济的选择。

相反，如果你的应用服务经常需要应对诸如大促场景这样的突发流量，那么使用令牌桶算法进行限流往往会更适合，当然，这时你也得接受令牌桶策略实现的高复杂度。

总之，针对每个限流策略的特点，在具体业务场景合理使用，才能发挥它最大的价值。

好了，聊完了限流策略，接下来我们将视线切换到另一个维度，从限流位置的视角来探讨限流实践，这里想说明的核心理念是：**良好的限流应该是分层次的**。这就好比我国长江的防洪策略，有三峡这个“巨无霸”在上游作为挡板调控洪峰，下游有葛洲坝、溪洛渡这样的中型水电站进一步调节水量，再往下还有无数小型水库进行精细化控流，形成了全方位保护的机制。限流也是同样的道理。

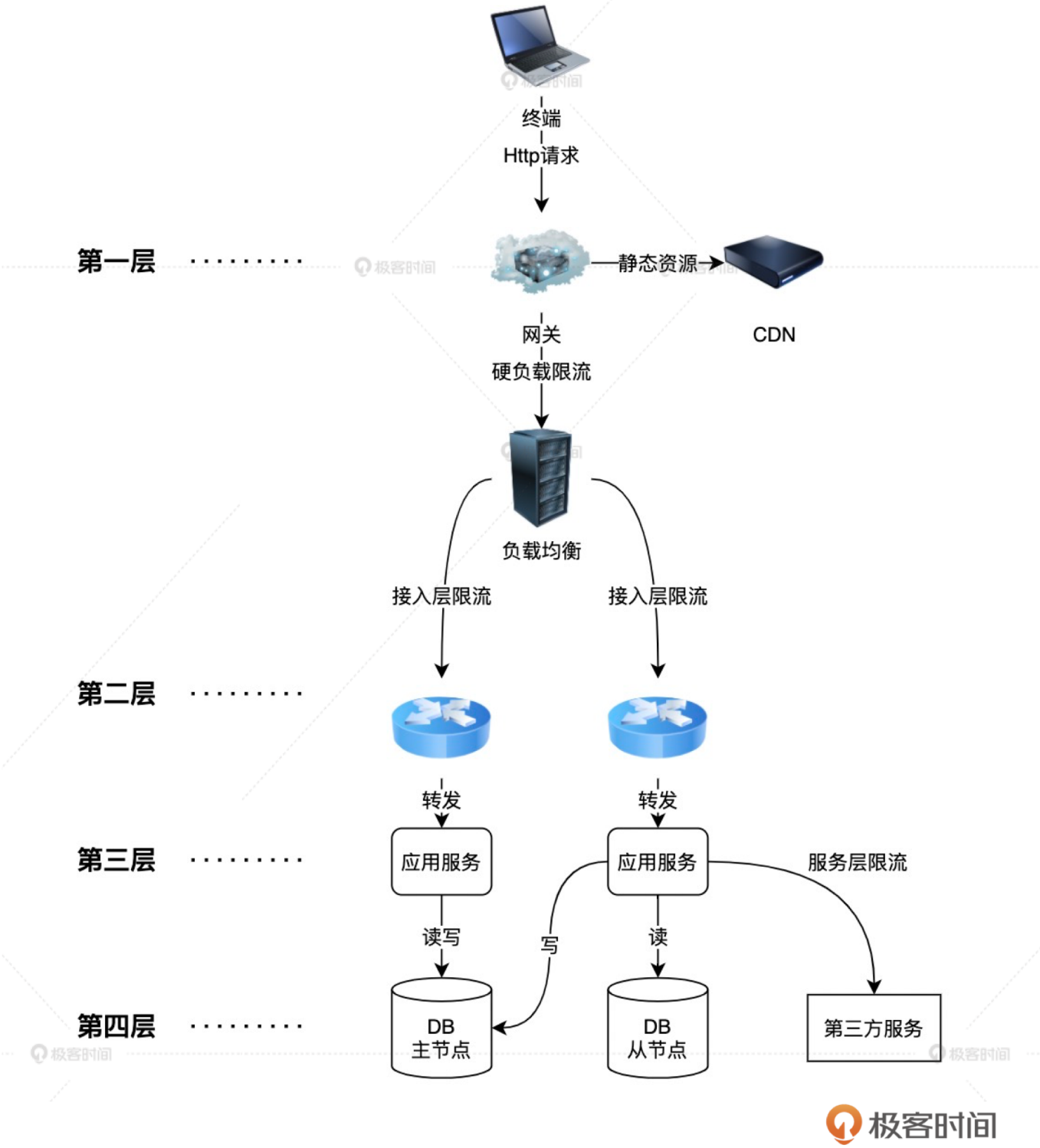
根据不同的限流位置，限流可以划分为**网关层限流、接入层限流、应用层限流、数据库层限流等**。我们直接来看下面这张图，图中自上而下描述了服务的流量走向，非常清晰地体现出层次化的限流思路，每个位置都恰如其分地加入了相应的限流策略，下面我具体解读一下。

第一层，入口网关可以针对域名或 IP 进行粗放式限流，静态资源直接走 CDN（内容分发网络），同时在这一层还可以将一些不合法的请求拦截掉，不要流入下游，仅放过合法请求。

第二层，硬负载和接入层都可以实施负载均衡和限流措施，分担服务压力。它的粒度比入口网关层更细，但还没有办法对单个服务实例进行限流。

第三层，到了应用服务层，每个服务可以有自己的单机或集群限流，调用第三方服务时，也可以单独进行限流。

第四层，与数据库的交互，采用读写分离的方式，分流数据库压力，同时也可以针对数据库读写请求进行限流。



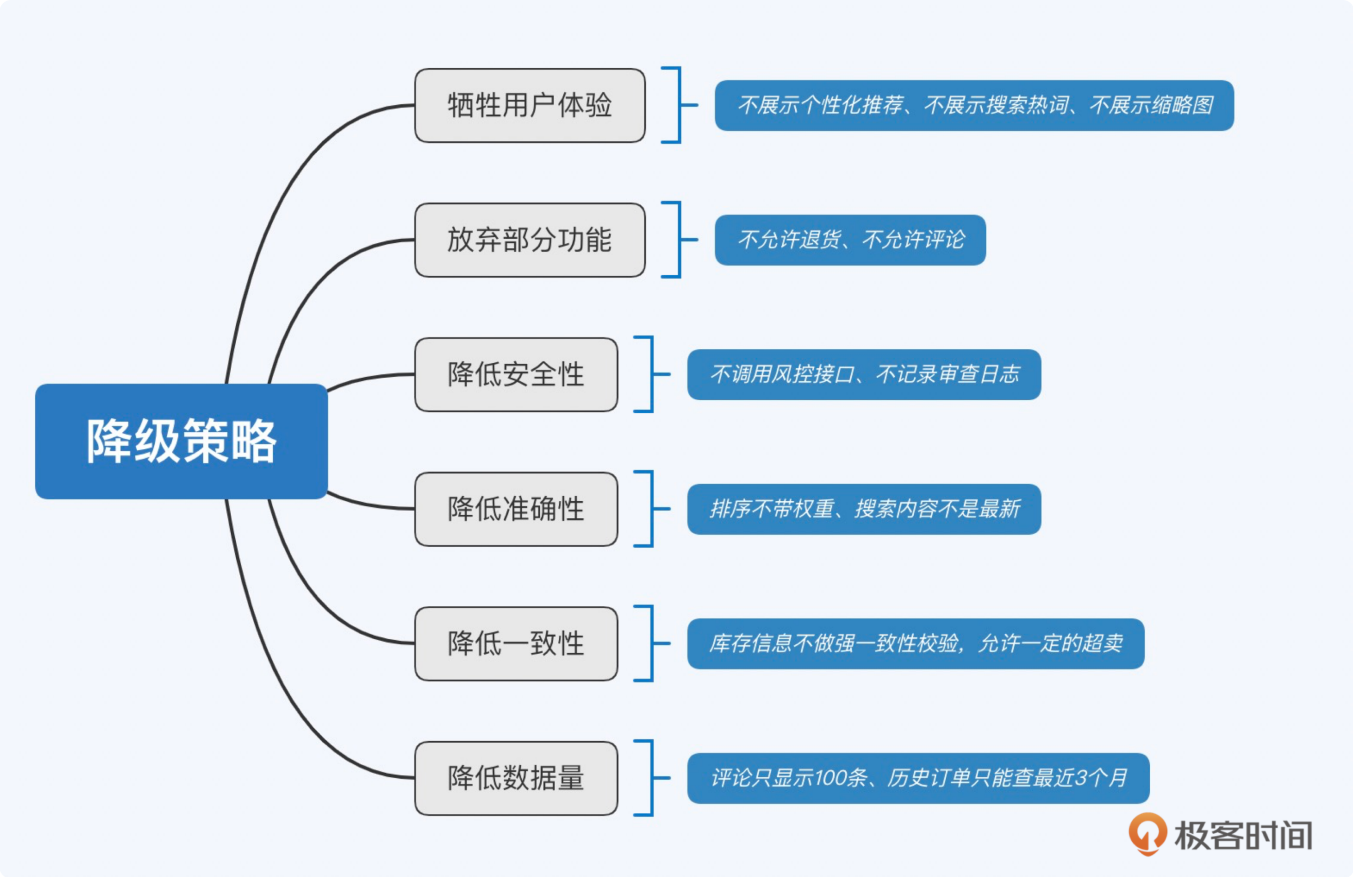
总之，在制定每一层的限流策略时，**都应该抱着不信任上层限流的思维**，这样即便某一层限流机制发生问题，也不至于引发全局问题，最终形成的限流体系才是最健壮、最可靠的。

总结一下，限流的核心思路是：第一，根据业务特点，比如是否有突发流量、输出流量，是否需要整形、是否需要平滑限流等，选择合适的限流策略，确保限流策略的健壮性和可靠性；第二，分层次，在不同的位置进行限流，多管齐下全方位完善限流体系。

降级的实践要点

前面提到的限流措施，是通过拒绝一部分服务流量来强行控制服务负载的手段，是从控制流量的视角来保证服务容量安全的；而降级则是从系统功能角度出发，**人为或自动地将某些不重要的功能停掉或者简化，以降低服务负载，这部分释放的资源可以去支撑更核心的功能**。简而言之，弃卒保帅。

降级在互联网行业的应用非常广泛，比如某大型电商在双 11 当天会将退货的功能降级，以确保核心交易链路的容量充足；某生鲜公司在遇到流量高峰期时，会将一部分个性化推荐的功能降级，以确保导购链路的工作正常，等等。抽象总结可以有以下几类策略，我同时提供了一些案例供你直观地理解它们。

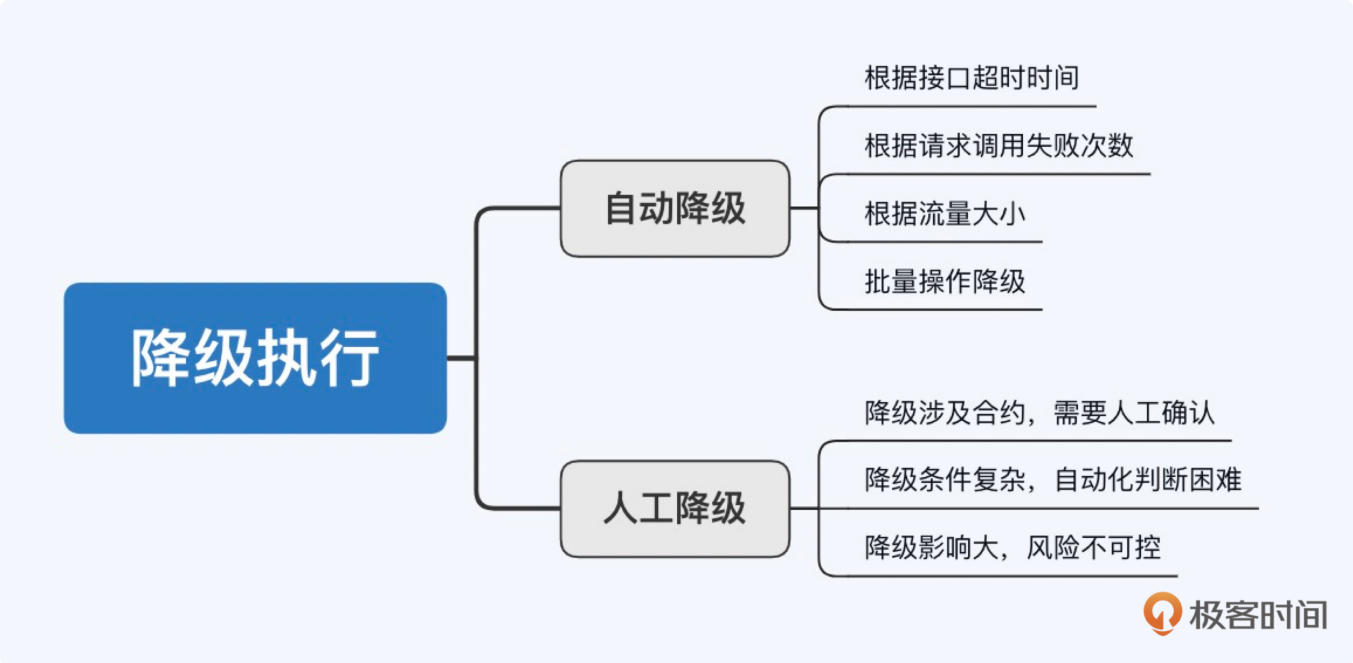


降级的技术实现本质上不难，也是微服务系统的常见功能，绝大多数情况下，是通过设置开关，并将开关收口至配置中心的方式作集中式管理的。我想深入讲解的知识点是，如果你已经有了一套降级系统，该如何管理上面的众多降级开关，把它们真正有效地推行下去，在服务容量发生风险时，及时止损，或提前止损。

首先，降级需要平衡好自动触发和人工执行两种做法。根据预置的触发条件自动执行降级固然快速便捷，但某些场景的降级触发条件其实挺难判断的，比如在系统偶发抖动的情况

下，到底是降还是不降，需要根据当时的业务情况做综合判断，这时候还是人工介入更靠谱。此外，还有一些降级会涉及到与第三方的合约，比如对支付宝、微信支付这类功能的降级，需要与对方确认后才能执行。

自动降级比较适合触发条件明确可控的场景，比如请求调用失败次数大于一定的阈值，或是服务接口超时等情况；还有一些旁路服务，比如审查日志记录等，如果压力过大也可以直接触发自动降级。我将这些要点总结为下面这张导图，供你参考。



其次，降级需要进行分级，因为降级操作都是有损的（如果无损，那你应该考虑一下被降级的功能是否还有存在的必要了），所以如果进行“温柔的”降级已经能够释放足够的容量，就没有必要过度降级。

基于这个理念，我们可以根据对业务的影响程度，制定降级的分级策略。比如在导购链路上，针对个性化推荐和滚动热词功能的降级就属于不太影响用户使用的范畴，可以定为 1 级降级；而不展示商品图片和评价内容这类降级，明显会对用户使用造成不便，应定为 2 级降级。在真实发生容量问题时，可以先执行 1 级降级，如果服务恢复，则无需执行 2 级降级。

最后，降级需要演练，而且是频繁的演练。有些服务的降级逻辑由来已久，随着服务代码的迭代更新，这些降级逻辑可能已经失效了，一旦服务出现问题真要降级时，这可是要命的。通过高频演练可以及时暴露这些无效的降级开关，防患于未然。

总结一下，**降级与限流有明显的区别，前者依靠牺牲一部分功能或体验保住容量，而后者则是依靠牺牲一部分流量来保住容量。** 一般来说，限流的通用性会更强一些，因为每个服务理论上都可以设置限流，但并不是每个服务都能降级，比如交易服务和库存服务，就不可能被降级（没有这两个服务，用户都没法购物了）。

降级的策略还是比较丰富的，因此需要从多个角度去化简。首先，将一部分判断条件简单的降级通过自动化手段去实现；其次，根据对业务的影响程度，对降级进行分级，达到有层次的降级效果；最后，通过高频演练，确保降级的有效性。

总结

这一讲，我主要介绍了容量治理的三个手段：扩容、限流和降级，将容量测试获得的数据和分析工作的价值真正落到实处，降低容量风险。

实际工作中，需要特别警惕无脑扩容，加强服务架构优化意识和根因分析能力；在扩容时，也应全面评估系统资源情况，做到资源平衡。

奥卡姆剃刀定律告诉我们：“如无必要，勿增实体”，简单的才是最有效的。限流策略的制定应充分考虑业务场景特征，选择最简单和最适合的限流策略，而不是盲目追求复杂的策略。同时，限流应当从全局视角看待，建立层次丰富可靠的限流体系。

降级的本质是为了解决服务资源不足和访问量增加的矛盾，而放弃一部分功能或体验。降级应当平衡好自动化降级和人工降级的关系，并通过分级的方式保证最小可用降级，最后辅以大量高频的验证，确保降级的有效性和熟练度。

容量保障，任重道远。没有无死角的容量治理手段，我们应当全方位、体系化地思考认识，多角度覆盖容量保障的方方面面。

课后讨论

现在有这样一个业务场景，它的主要工作是审阅、处理和发放工资。我们有一个服务系统支撑这个业务场景，其中，审阅和处理工资是由公司财务在系统上完成的，在每月的最后一周是高峰期，其他时间几乎没有请求；而发放工资的功能是对接银行的第三方接口实现的，这个接口非常脆弱，每秒只能处理 5 个请求。

请问，针对这样的一个服务系统，选取哪种限流策略比较合适？欢迎与我交流你的想法，在评论区给我留言。如果你觉得这一讲对你有帮助的话，欢迎把它分享给你的朋友。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | 容量指标分析经典5问：响应时间真的是越短越好吗？

下一篇 05 | 全链路压测：系统整体容量保障的“核武器”（上）

精选留言 (2)

写留言

Roy Liang
2021-05-20

我觉得这个系统是内部服务，即使高峰期流量也有限，限流只针对第三方接口，使用简单的固定窗口就可以了

展开

作者回复: 虽然是内部系统，但是如果工资没及时发出去也挺可怕的吧，哈哈。可以再想一想，针对这里的第三方接口限流，有没有流量整形和平滑限流的要求？

继贤
2021-05-20

老师，咨询下，对于一些公共服务，会同时支撑好多个产品，如果按照每个产品每个场景业务指标来规划，可能公共服务的规划容量会很大，实际应用中，所有产品并不会同时出现峰值的情况，那么问题来了，对于公共服务，如何做成本最优估算？

展开

作者回复: 好问题！我来简化一下问题，考虑有一个公共服务A，支撑了业务服务B和C。B和C不一定会在某个时间点同时出现峰值，但是综合起来对于到达公共服务A的总压力来说，一定会有一

个峰值点X，在这个峰值点X上B和C不一定是峰值，但它们对于A的累计压力值是最高的。于是，在X这个峰值点的业务场景就是你应该考虑的容量预估的基准场景。

当然，前提是B和C要保证能够承载各自的峰值压力。